**We Claim:**

1. A method of providing Web access to remote program, comprising the steps of:

automatically generating a Web application from a description of an interface of the program;

storing the Web application on a server accessible to a client;

transmitting the Web application to the client in response to a request by the client;

receiving an indication from the client for invoking the interface;

invoking the interface;

receiving a result from the program in response to invoking the interface

transmitting a response Web application to the client based on the result.

2. The method of claim 1 wherein the program is CORBA-compliant.

3. The method of claim 1 further including the step of generating the Web application from the description of the interface with nested Web pages such that the interface corresponds to an interface Web page and an operation within the interface corresponds to an operation Web page, the interface Web page comprising a link to the operation Web page.

4. The method of claim 3 wherein the nested Web pages include a Web page for submitting data to invoke the interface.

5. The method of claim 4 further including the step of customizing one of the automatically generated nested Web pages to change the appearance of at least one automatically generated Web page.

6. The method of claim 4 further including the step of customizing the automatically generated nested Web pages by performing one act from a set consisting of adding an element, removing an element, removing a javascript element, replacing a form field with a fixed value, and preventing display of an element, in at least one of the automatically generated Web pages.

7. The method of claim 1 wherein the automatically generated Web application includes at least one web-page.

8. The method of claim 6 further including the step of customizing the nested Web pages in response to adding a new operation to the interface.

9. A method for transforming data represented in a first markup language into a second markup language comprising the steps of:

parsing the data to generate a parse tree reflecting the syntax of the data;

traversing the parse tree and annotating the parse tree by adding data to the parse tree representing an aspect of the data in the second markup language.

10. A method for invoking an object, comprising the steps of:

generating a description of the interface of the object;

generating metadata representing the interface of the object from the

description;

storing the metadata;

generating a representation of an invocation of the object in a markup language from the metadata;

transmitting the representation of the invocation to a client program configured to invoke the object by interpreting the representation;

receiving an invocation from the client program;

based on the metadata, interpreting the received invocation.

11. A method for transforming data represented in a first markup language into a second markup language comprising the steps of:

invoking a factory object to obtain an instance of an product object conforming to an interface for generating a representation in the second markup language;

parsing the data;

invoking the product object with information derived from the parsed data to generate data representing an aspect of the data in the second markup language.

12. The method of claim 10, further comprising the steps of:

configuring the factory object to provide a different product object in response the invocation and an indication of a different second markup language.

13. A method for displaying a response to an invocation of a remote program, the response comprising data having a length unknown at the time of invocation, comprising the steps of:

transmitting a Web application comprising executable code for receiving a

portion of the response having a length not known at the time of invocation and generating HyperText Markup Language for representing the received portion of the response.

14. A method for invoking a remote program, comprising the steps of:

receiving information including identifiers describing an interface of the remote program;

checking the identifiers for name clashes with a scripting language;

based on the received information, generating script for execution by a Web browser including identifier names that do not clash with the scripting language.

15. A system for processing an XML message, comprising:

an XSLT processor for transforming XML messages;

a callback registry;

one or more callback objects registered in the callback registry that are selectively activated according to registration information stored in the callback registry to process XML messages.

16. A method for converting an invocation on a remote object in a markup language into a native call, comprising the steps of:

transmitting the invocation, in the markup language, to a remote location;

interpreting, in response to receiving the invocation at the remote location, the markup language based on metadata representing an interface on the object;

making the native call on the object via the interface.

17. In a software environment for providing web-access to at least one CORBA object via simple object access protocol (SOAP) in a Java encoded object, a method for integrating SOAP interactivity in Java, the method comprising the steps of:

issuing, by a developer, an application program interface (API) call to cause construction of a request object in accordance with an operation and any required parameters for the operation;

sending a SOAP message containing a name for the operation and the required parameters for the operation to a target object via a SOAP processing server program;

creating a reply object based on executing the operation at the target object;

receiving a response, based on the reply object, from the SOAP processing server program.

18. The software environment of claim 17 wherein the reply object is created automatically in response to receiving the response from the SOAP processing server program.

19. The software environment of claim 18 wherein the Java encoded object is in a Java Server Page.